

FIG.1

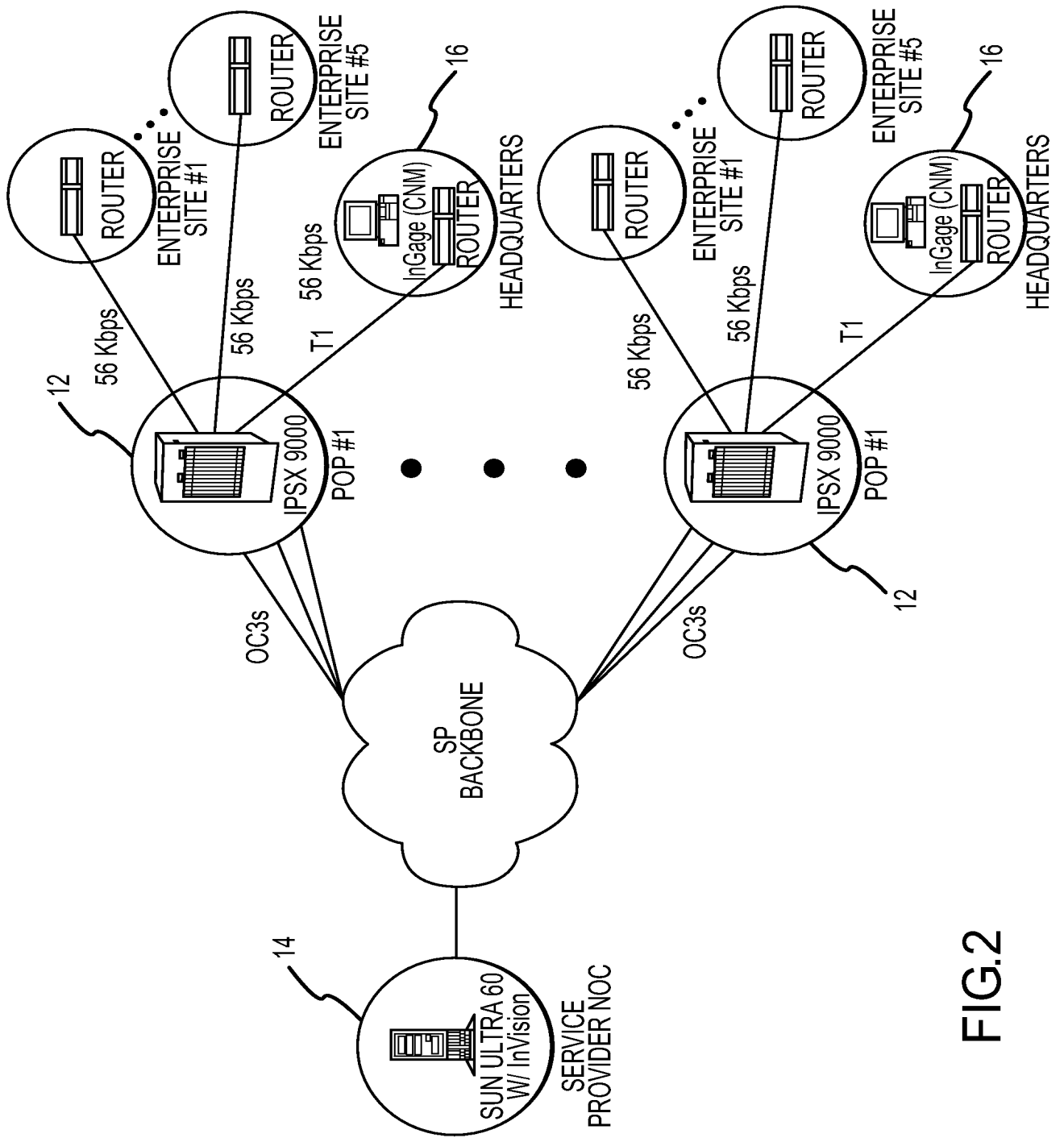


FIG.2

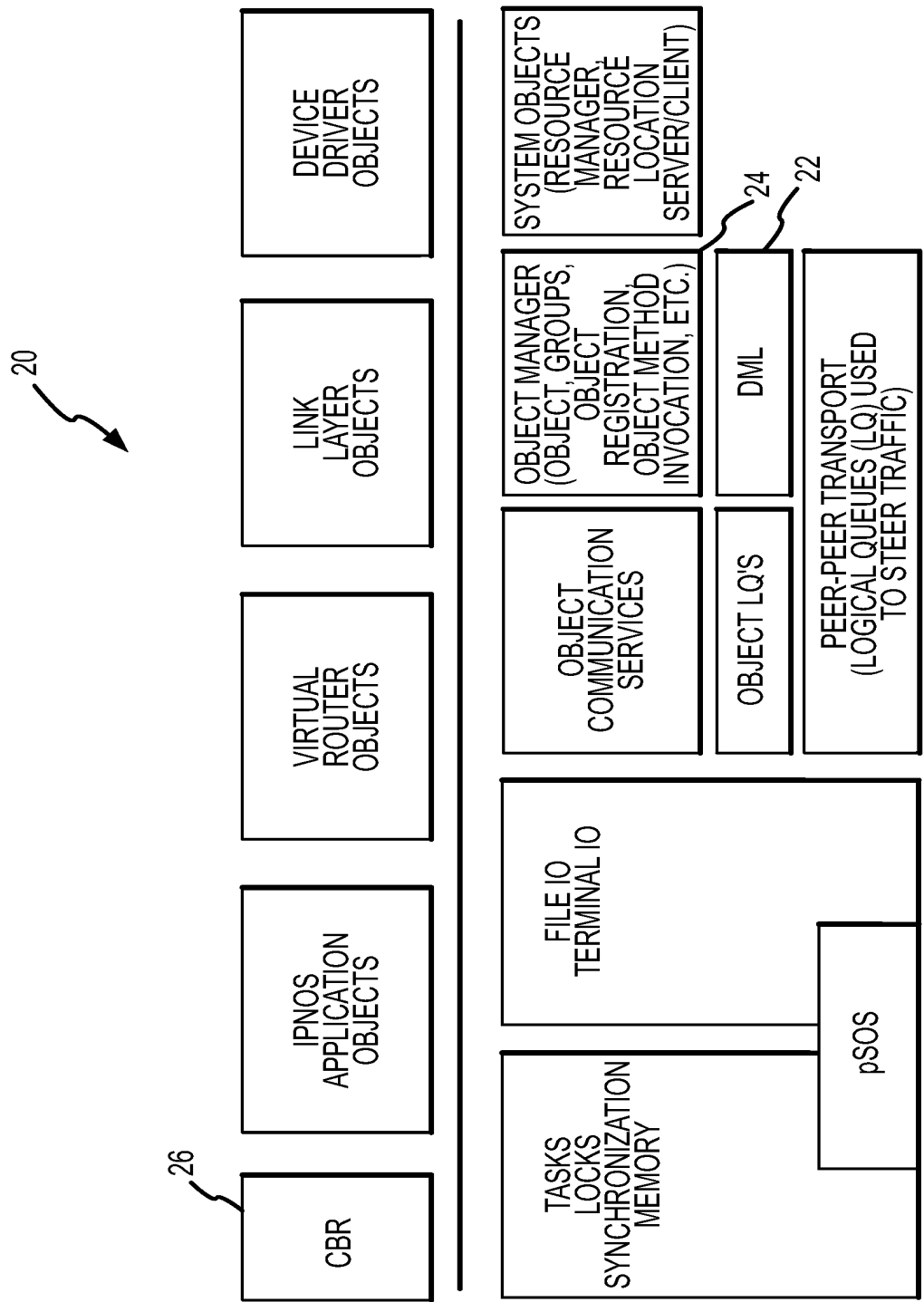


FIG.3

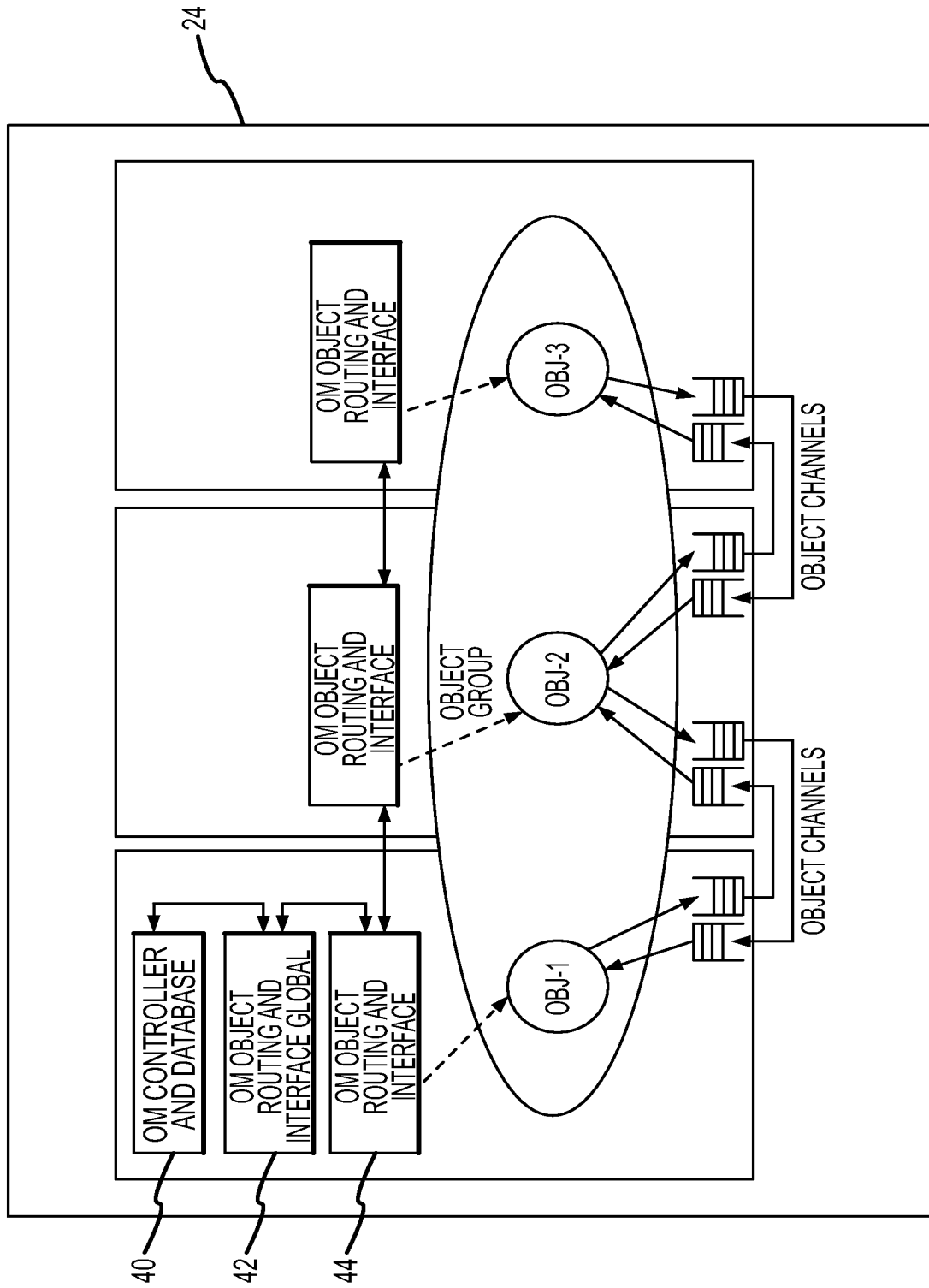


FIG.4

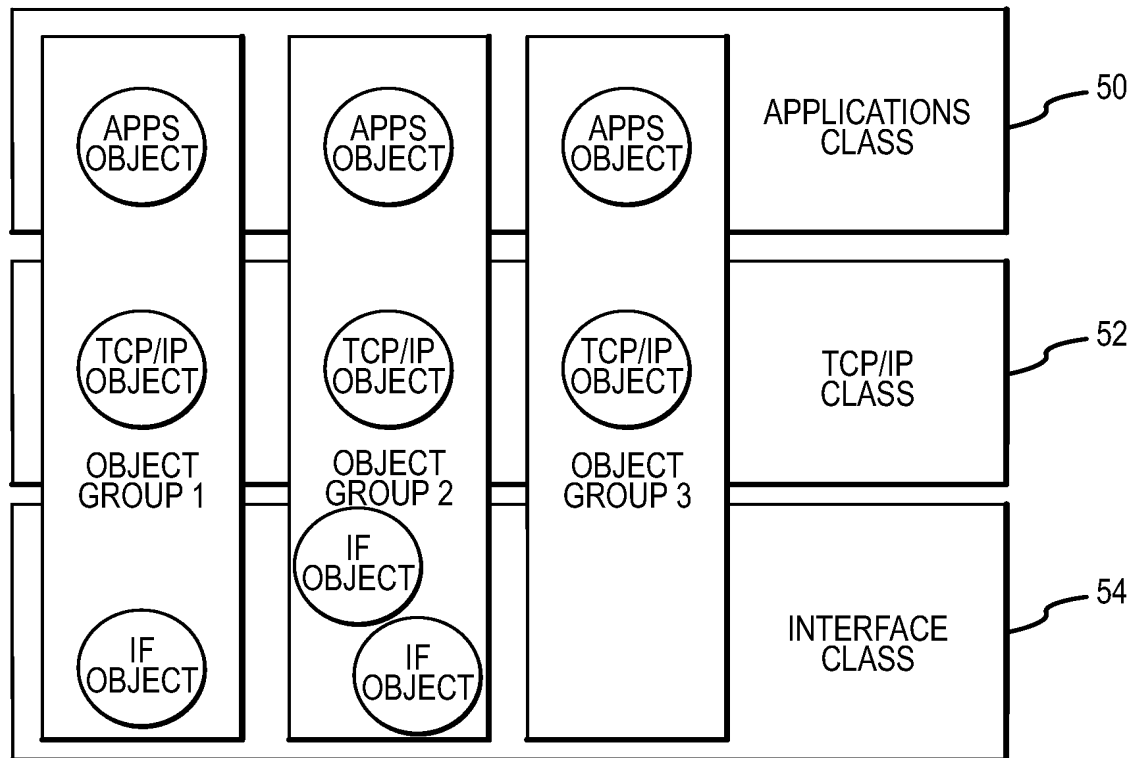


FIG.5

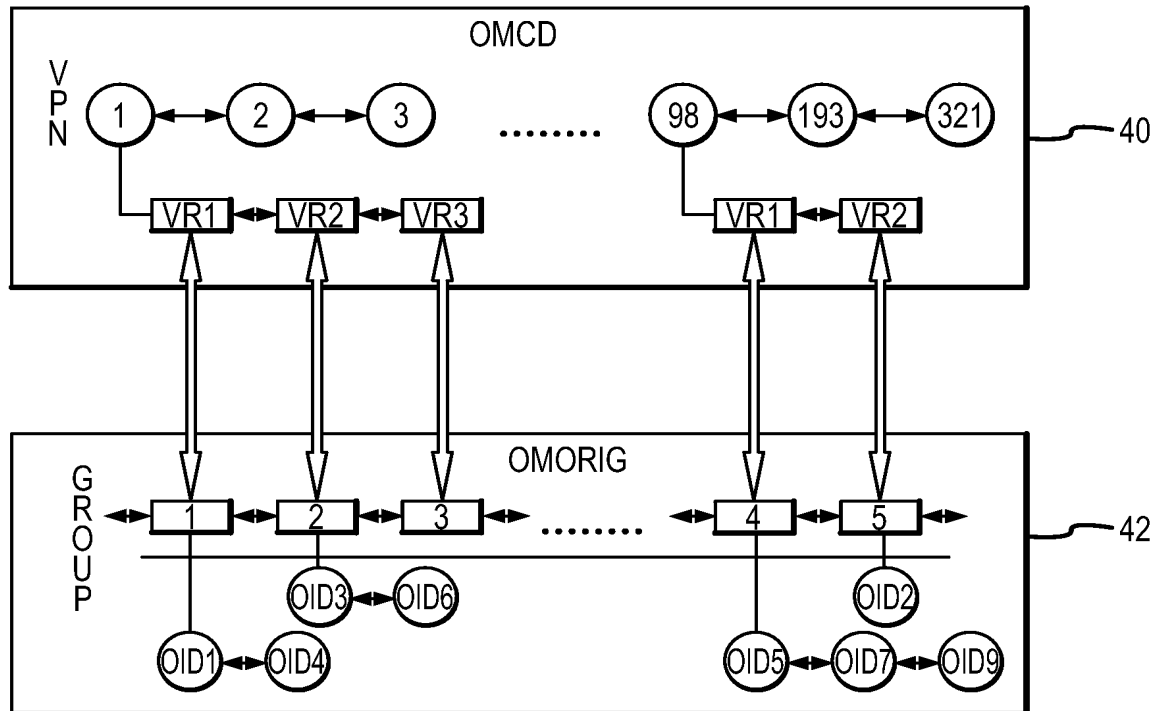


FIG.6

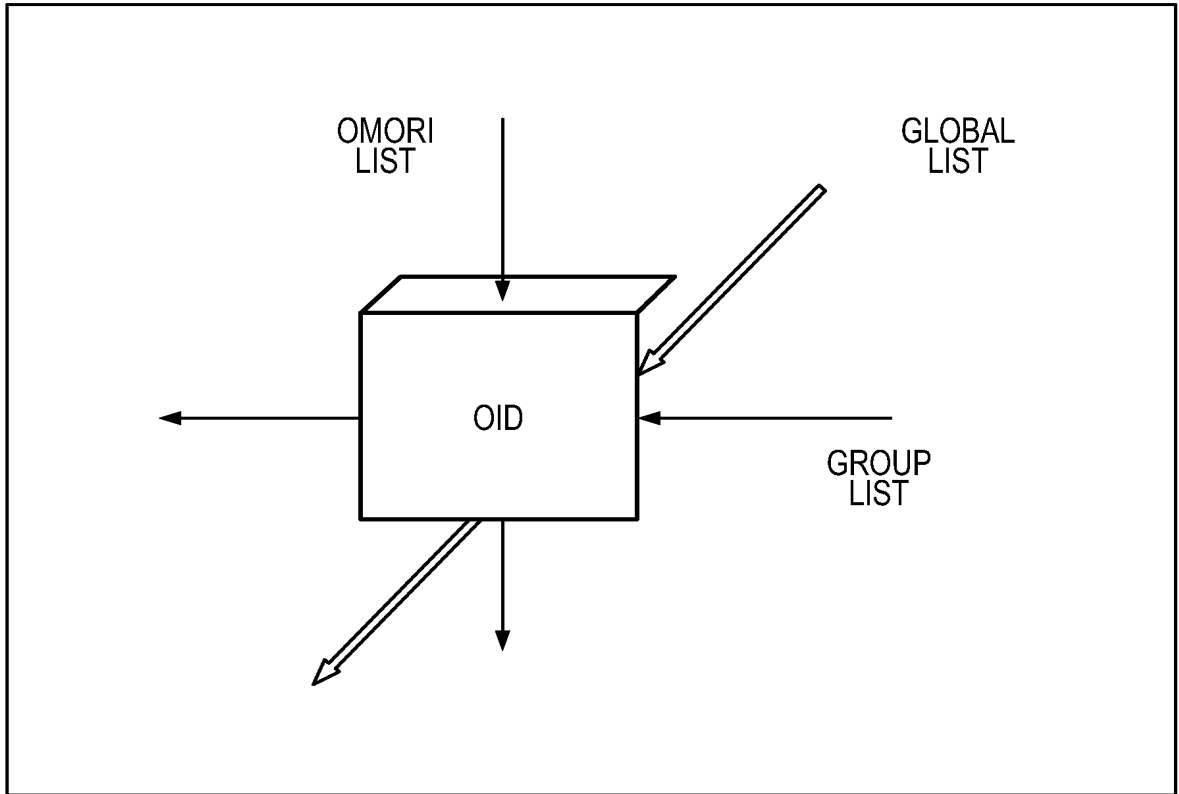


FIG.7

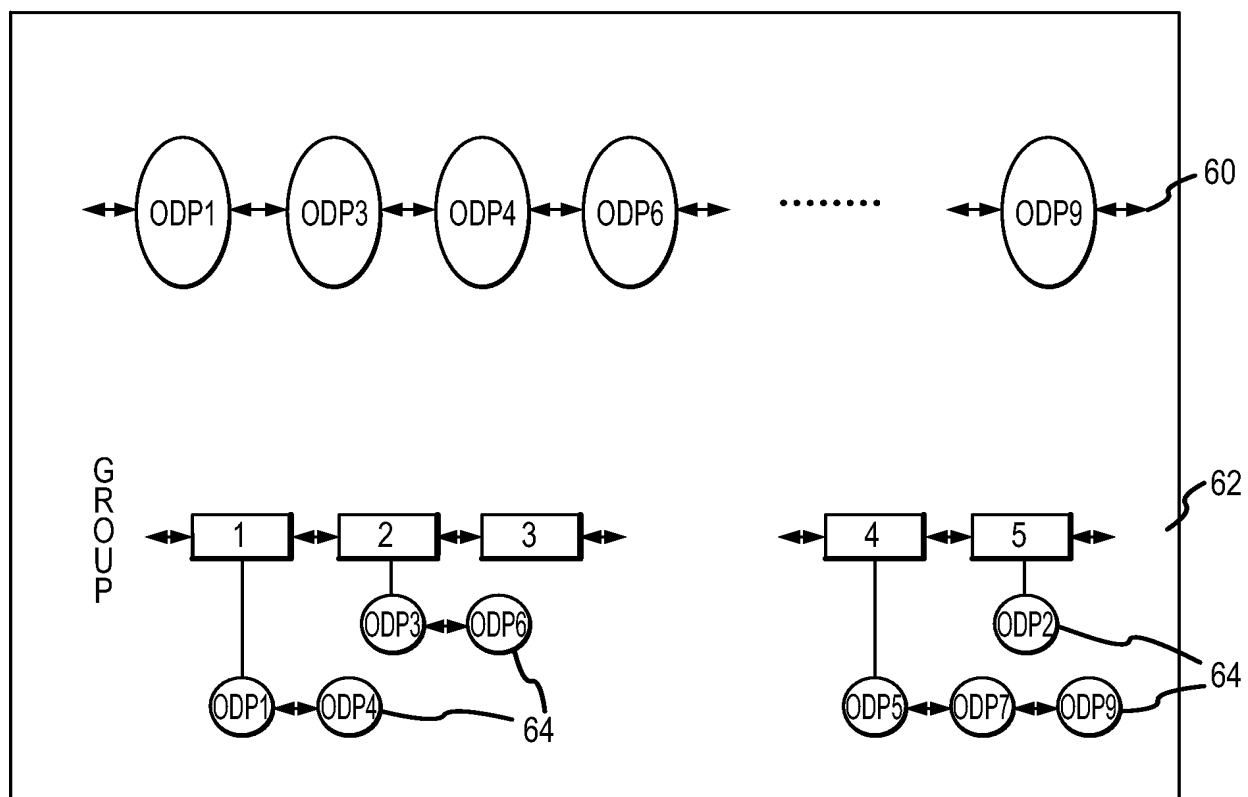


FIG. 8

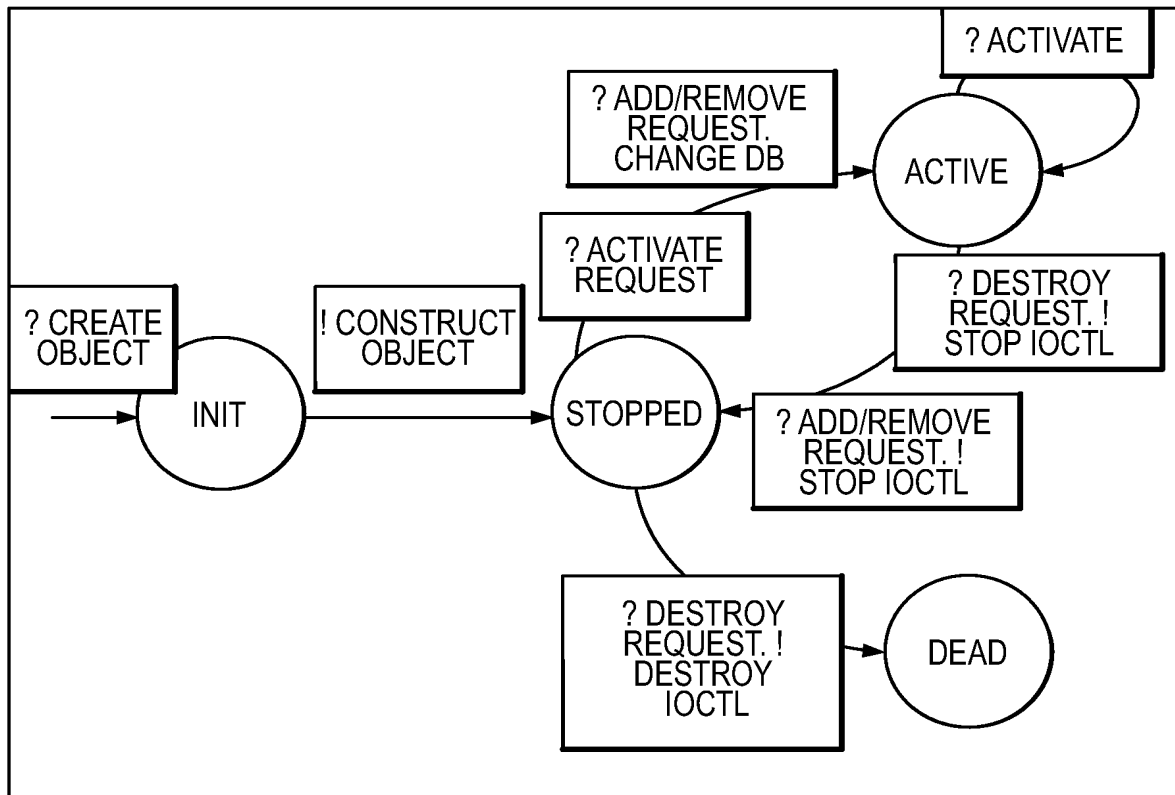


FIG.9

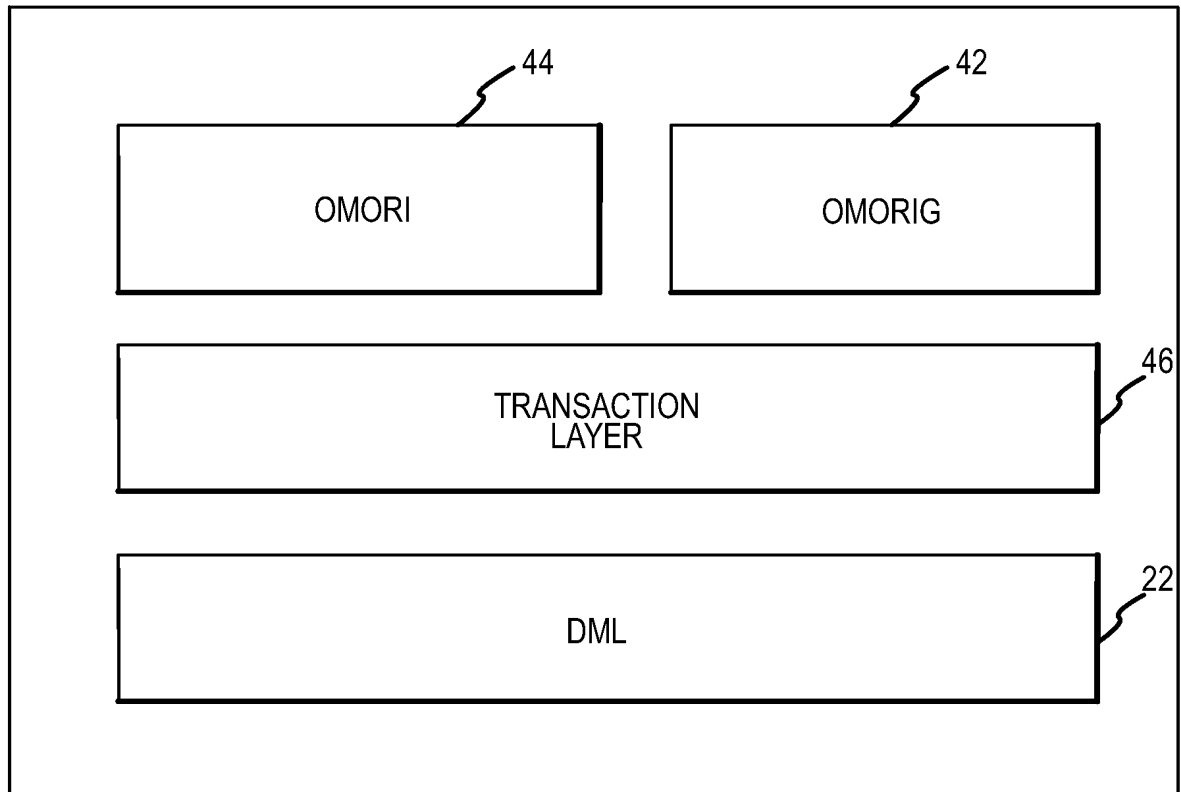
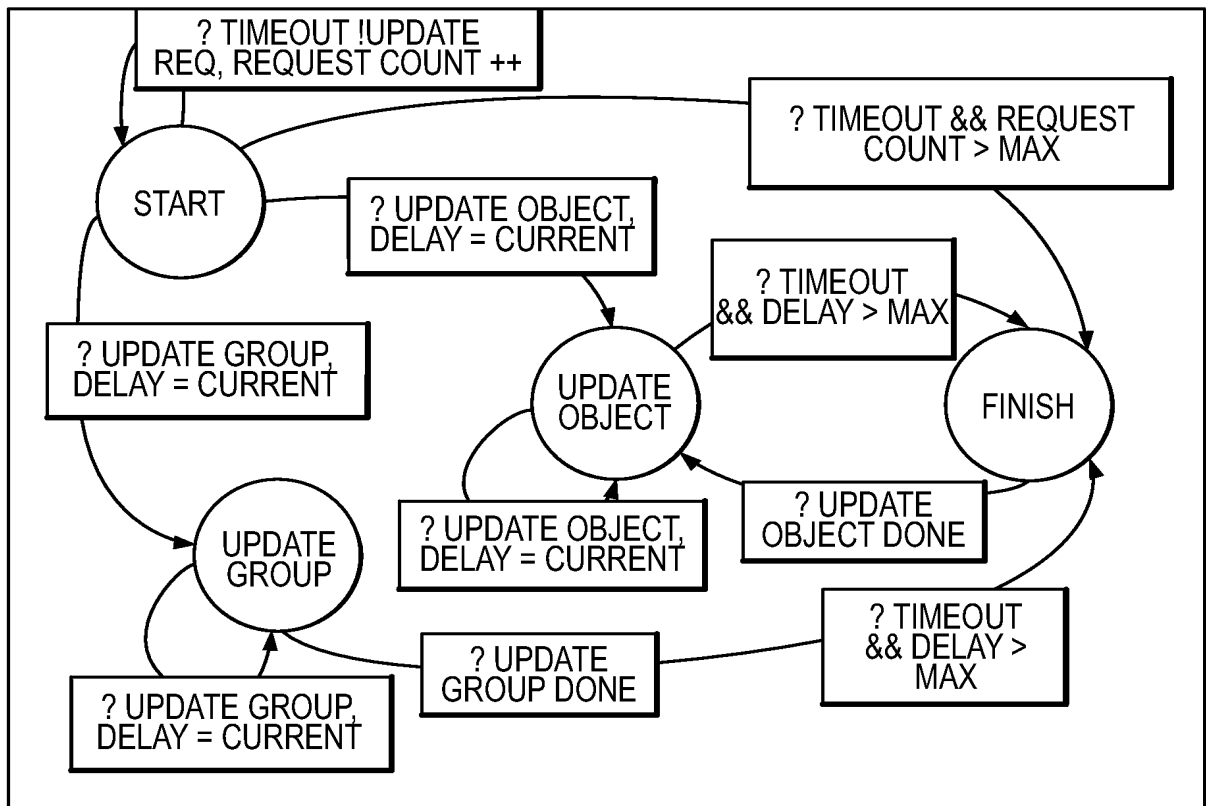


FIG.10



STATE	EVENT	ACTION
START	TIMEOUT && (REQUEST COUNT < MAX)	SEND UPDATE REQUEST
START	TIMEOUT && (REQUEST COUNT > MAX)	PEER DID NOT REPLY. UPDATE FAILED TRANSIT TO FINISH STATE.
START	RECV UPDATE GROUP MESSAGE	TRANSIT TO UPDATE GROUP STATE. SET LAST UPDATE EQUAL TO THE CURRENT TIME.
START	RECV UPDATE OBJECT MESSAGE	TRANSIT TO UPDATE OBJECT STATE. SET LAST UPDATE EQUAL TO THE CURRENT TIME.

FIG.11

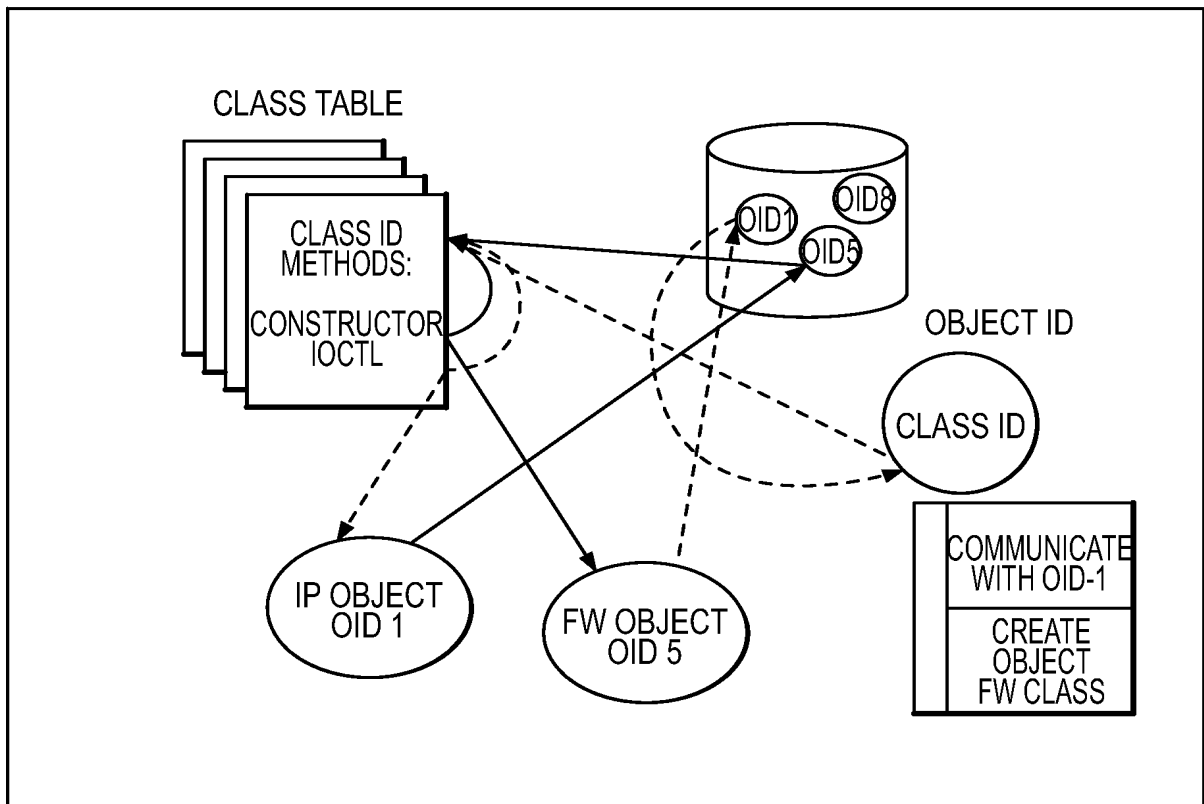


FIG.12

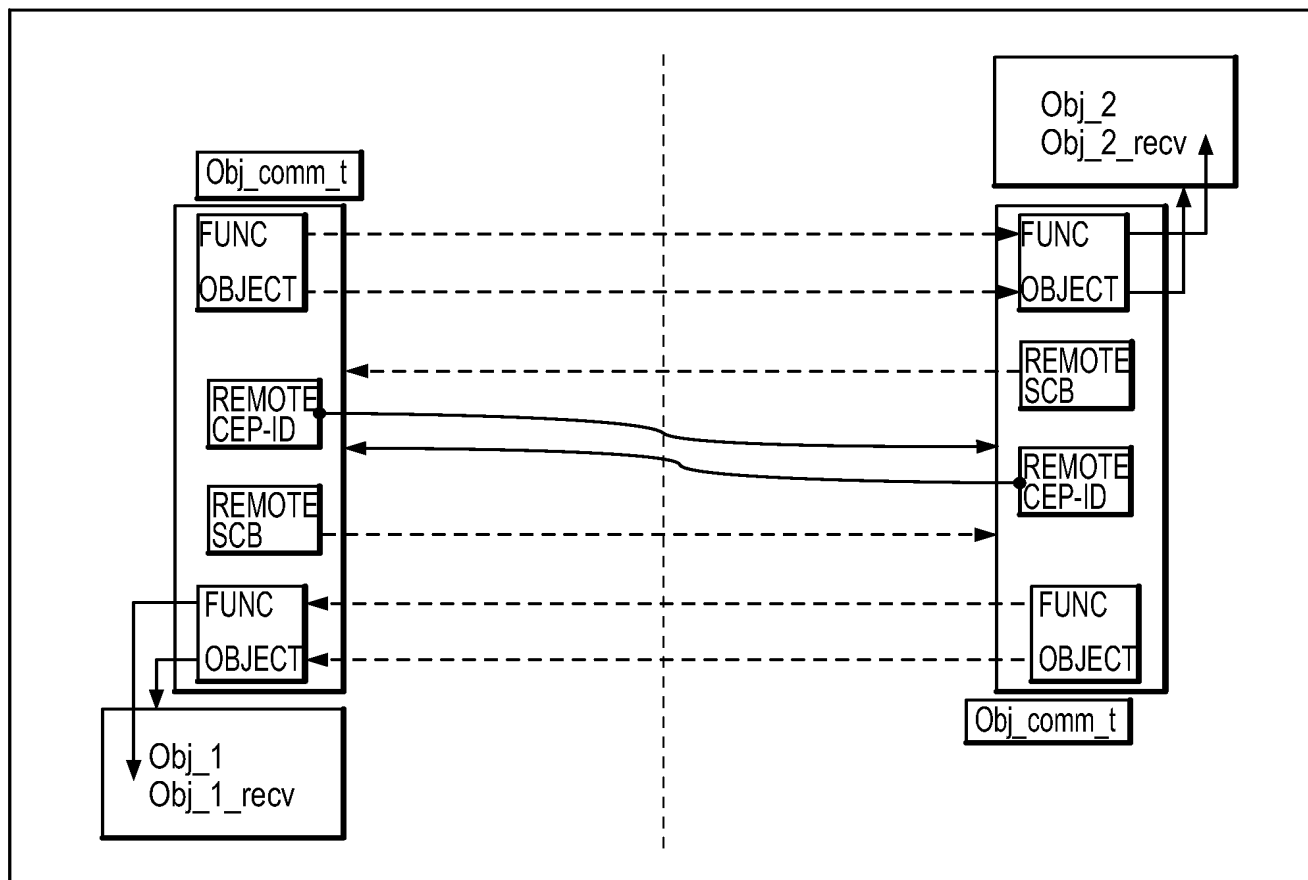


FIG.13

FIG.15

STEP	LOCAL CEP OBJECT	LOCAL IPNOS	LOCAL RM/LQ	REMOTE RM/LQ	REMOTE IPNOS	REMOTE CEP OBJECT
1	obj_associate_channel(local_chan, local_cep_id, remote_cep_id)					
2		/* Allocate remote LQ */ resmng_alloc_resource (RESOURCE_DATA_CONNECTION, 0, remote_cep_id -> object.address_space_id, &remote_lq)				
3				Lookup resource tag and allocate from remote LQ		
4		/* Ask remote LQ to allocate local LQ*/ status = omori_obj_ioctl_by_id (&remote_lq, remote_lq.group, OBJ_CTL_CODE_ANY (LQUSER_BIND), &lq_bind, sizeof (lq_bind)); memcpy (&local_lq,				

FIG.16a

		&lq_bind.lq_object.local, sizeof (object_id_t));				
5				Use resmgng_alloc_resource0 to allocate <i>local</i> LQ		
6			Lookup resource tag and allocate from <i>local</i> LQ			
7				Return allocated <i>local</i> LQ		
8		/* Bind Local and Remote LQs*/ status = omori_obj_ioctl_by_id (&local_lq, local_lq.group, OBJ_CTL_CODE_ANY (LQUSER_BIND), &lq_bind, sizeof (lq_bind));				
9			Setup LQ-API parameters to point to <i>remote</i> LQ	Setup LQ-API parameters to point to <i>local</i> LQ		
10		/* Push local LQ as a service onto local channel*/ status = omori_obj_ioctl_by_id				

FIG.16b

		(&local_lq, local_lq.group, OBJ_CTL_C ODE_ANY (LQUSER_BI ND), &lq_bind, sizeof (lq_bind));				
11	Lookup CEP address					
12					<i>/* Push remote LQ as a service onto remote channel*/</i> status = omori_obj_ioc tl_by_id (&remote_lq, local_lq.group, OBJ_CTL_C ODE_ANY (LQUSER_BI ND), &lq_bind, sizeof (lq_bind));	
13						Lookup CEP address

FIG.16c

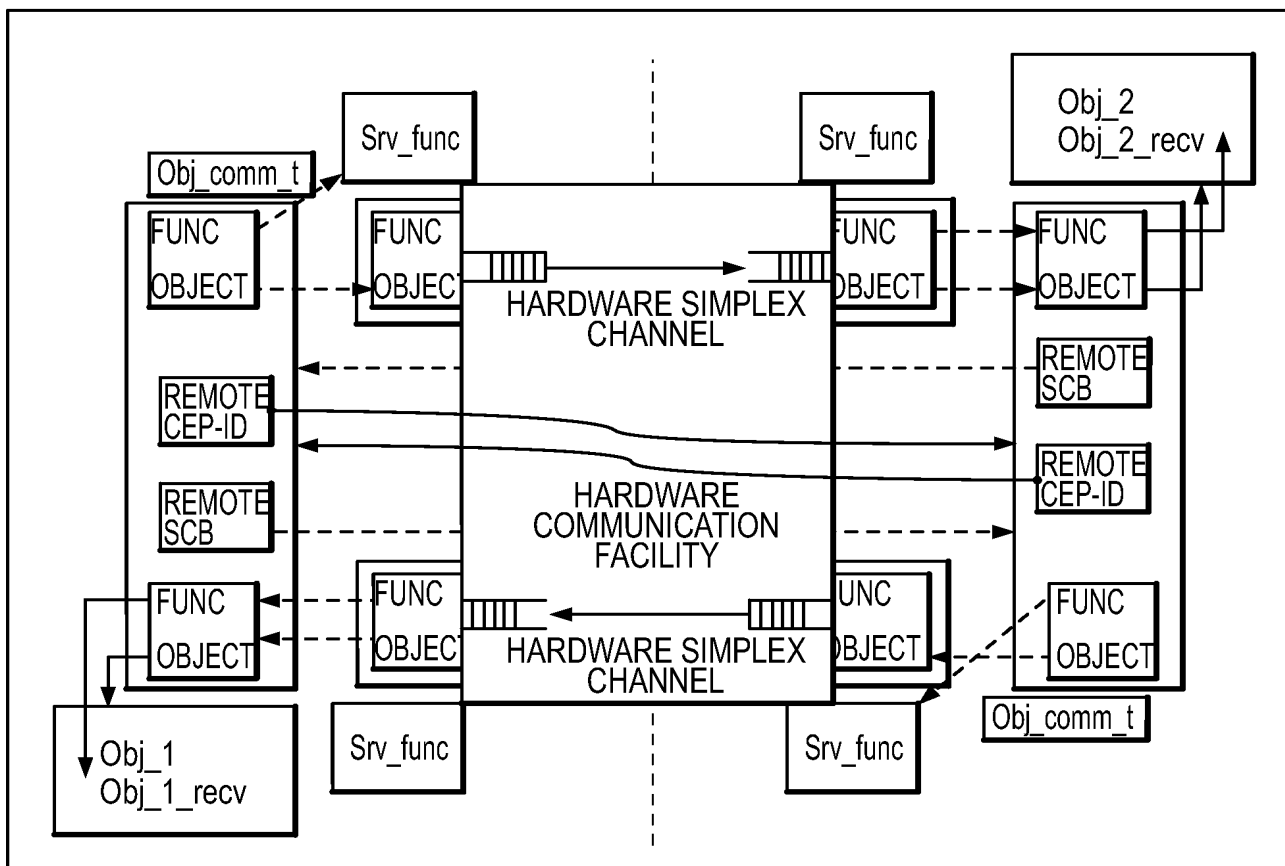


FIG.17

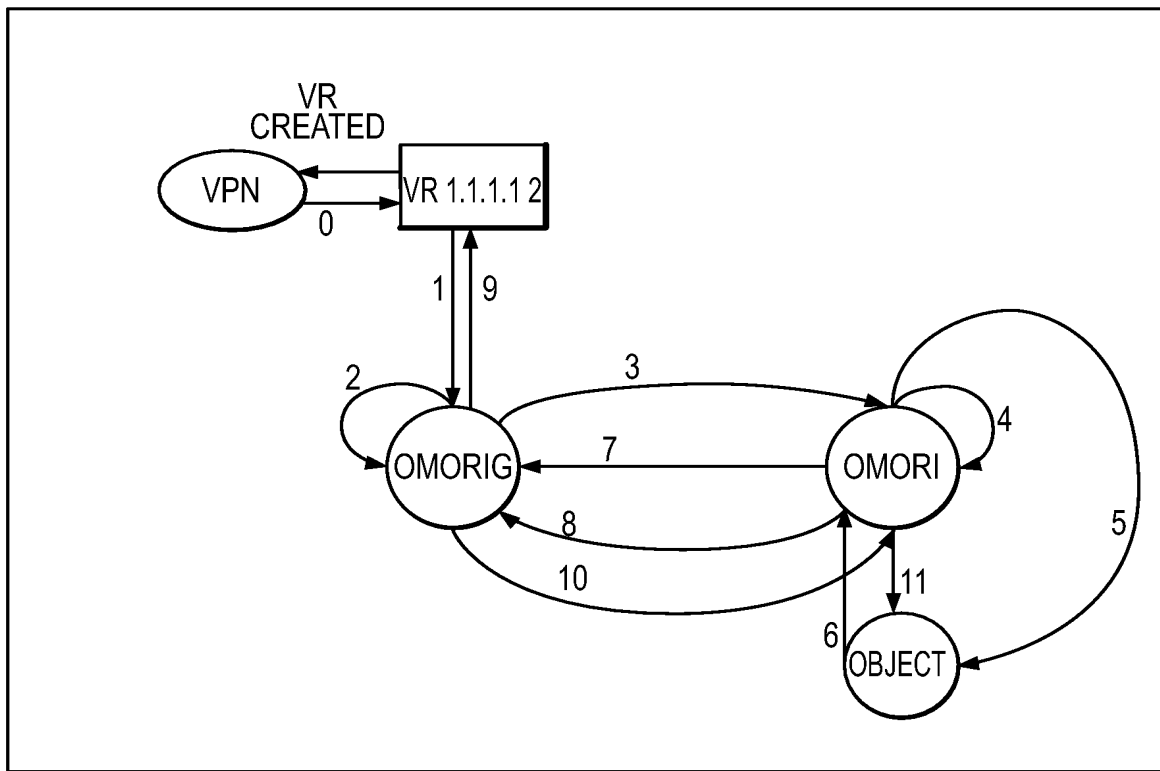


FIG.18

STEP	OMCD	OMORIG	OMORI	OBJECT
0	Create unique vr_descriptor t for specified VPN, fills with default value and adds VR to the list of VR per VPN.			
1	Requests group creation for specified VR with class_group_selector on specified address space.			

FIG.19a

2		Create group 1; create object id link of selected class. Validate address space id on capability to service specified object class. Send request CREATE OBJECT to capable OMORI (2). Wait for OMORI reply.		
3			Receive CREATE OBJECT request for specified group. Lookup a group; not found. Create group 1; Create object descriptor of selected class. Call the class constructor.	
4		Receive MV_OBJ_TO_GROUP request; add object id to OMORIG Database	add object to the group, send MV_OBJ_TO_GROUP request to OMORIG	
5				Create and initialize an object; return SUCCESS or FAILURE.
6			If FAILURE remove object from the group, send MV_OBJ_TO_GROUP and RM_OBJ_FROM_GROUP to OMORIG; =====	
			Else send reply for CREATE_OBJECT request to OMORIG	
7		Receive MV_OBJ_TO_GROUP request, move object to the group O(OM_BASE_GROUP); Receive RM_OBJ_FROM_GROUP request; remove object id from OMORIG		

FIG.19b

		Database =====		
8		Receive Object CREATE reply. Signal to OMCD		
9	VR created, return status to user			
10		Send ACTIVATE object message to OMORI (2)		
11			Receive ACTIVATE object message. For all the objects of this group send generic IOCTL ACTIVATE_OBJECT	Activate object, Do object-specific action to make it active, operational

FIG.19c

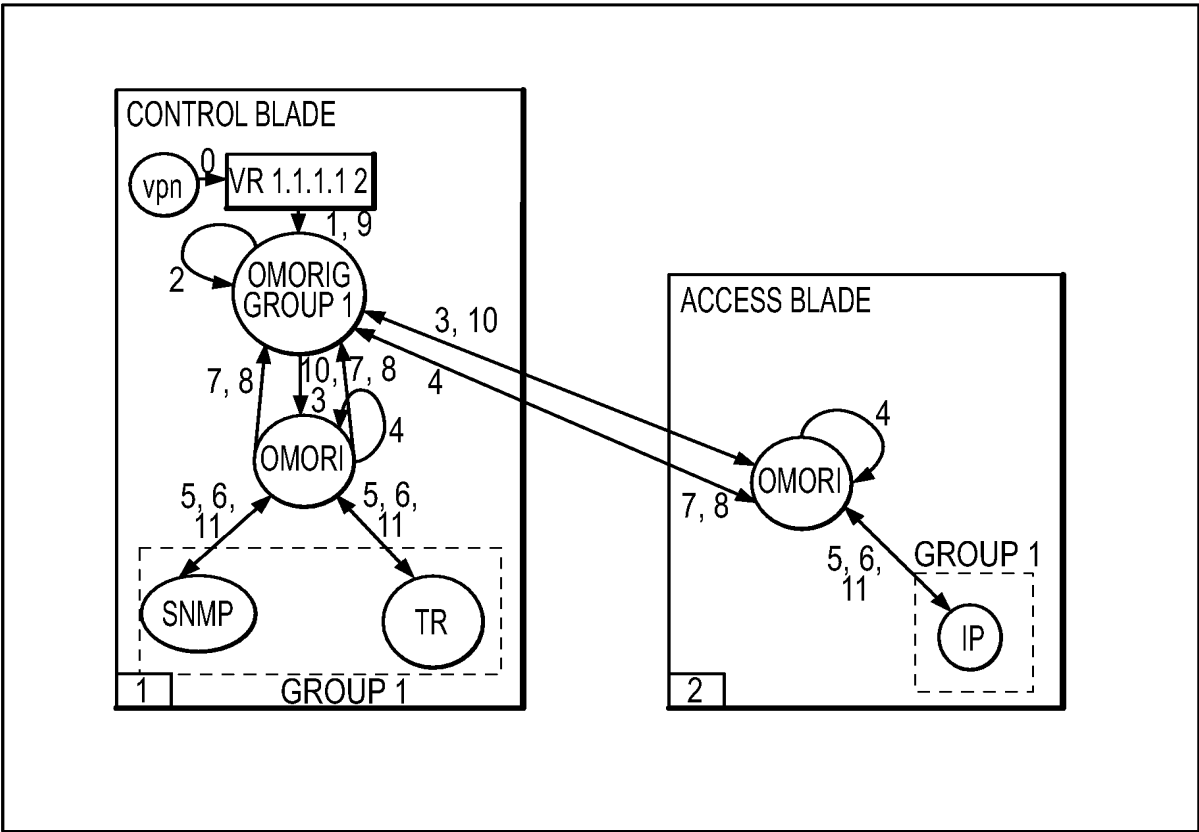


FIG.20

STEP	OMCD	OMORIG	OMORI	OBJECT
0	Create unique vr_descriptor_t for specified VPN, fills with default value and adds VR to the list of VR per VPN.			
1	Requests group creation for specified VR with class_group_selector on specified address space.			
2		Create group 1; create object id link of selected class. Validate address space id on capability to service specified object class. Send request CREATE_OBJECT to capable OMORIs (1 and 2). Wait for reply from both OMORIs.		
3			Receive CREATE_OBJECT request for specified group. Lookup a group; not found. Create group 1; Create object descriptor of selected class. Call the class constructor.	
4		Receive MV_OBJ_TO_GROUP request; add object id to OMORIG Database	add object to the group, send MV_OBJ_TO_GROUP request to OMORIG	
5				Create and initialize an object; return SUCCESS or FAILURE
6			If FAILURE remove object from the group, send MV_OBJ_TO_GROUP and RM_OBJ_FROM_GROUP to OMORIG;	

FIG.21a

			Else send reply for CREATE OBJECT request to OMORIG	
7		Receive MV OBJ TO GROUP request, move object to the group O(OM_BASE_GROUP); Receive RM OBJ FROM GROU P request; remove object id from OMORIG Database		
8		Receive Object CREATE reply. If all the object replied then signal to OMCD, otherwise do nothing		
8	VR created, return status to user			
10		Send ACTIVATE object message to every OMORI (1,2) where objects were created		
11			Receive ACTIVATE object message. For all the objects of this group send generic IOCTL ACTIVATE_OBJECT	
12				Activate object, Do object-specific action to make it active, operational

FIG.21b

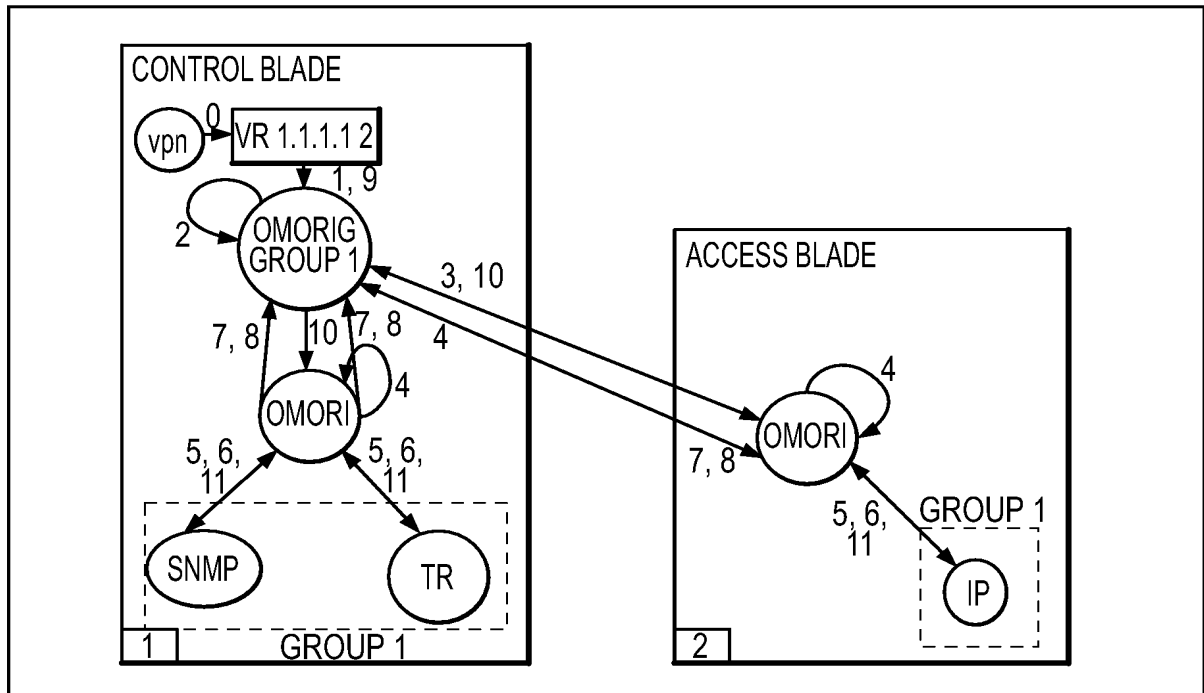


FIG.22

STEP	OMCD	OMORIG	OMORI	OBJECT
0	For specified VPN and VR lookup vr_descriptor. Call OMORIG to delete corresponding group.			
1		Lookup group descriptor by specified id. Filter OMORIs which have objects to be destroyed(which belong to the specified group)		
2			Receive DESTROY_GROUP_OBJECTS request for specified group. Lookup a group; Send generic IOCTL STOP_OBJECT to every local object, which belongs to the group	
3				Stop operating, destroy all connections with other objects

FIG.23a

4			Send generic IOCTL DESTROY_OBJECT to every local object, which belongs to the group	
5				Free itself
6			If FAILURE remove object from the group, send MV_OBJ_TO_GROUP and RM_OBJ_FROM_GROUP to OMORIG;	
7		Receive MV_OBJ_TO_GROUP request, move object to the group 0(OM_BASE_GROUP); Receive RM_OBJ_FROM_GROUP request; remove object id from OMORIG Database		
8		Receive Object DESTROY_GROUP_OBJECTS. Subtract number of destroyed objects from the total number of objects in the group (VR). If all objects destroyed then signal to OMCD, otherwise do nothing.		
8	VR destroyed, return status to user			

FIG.23b

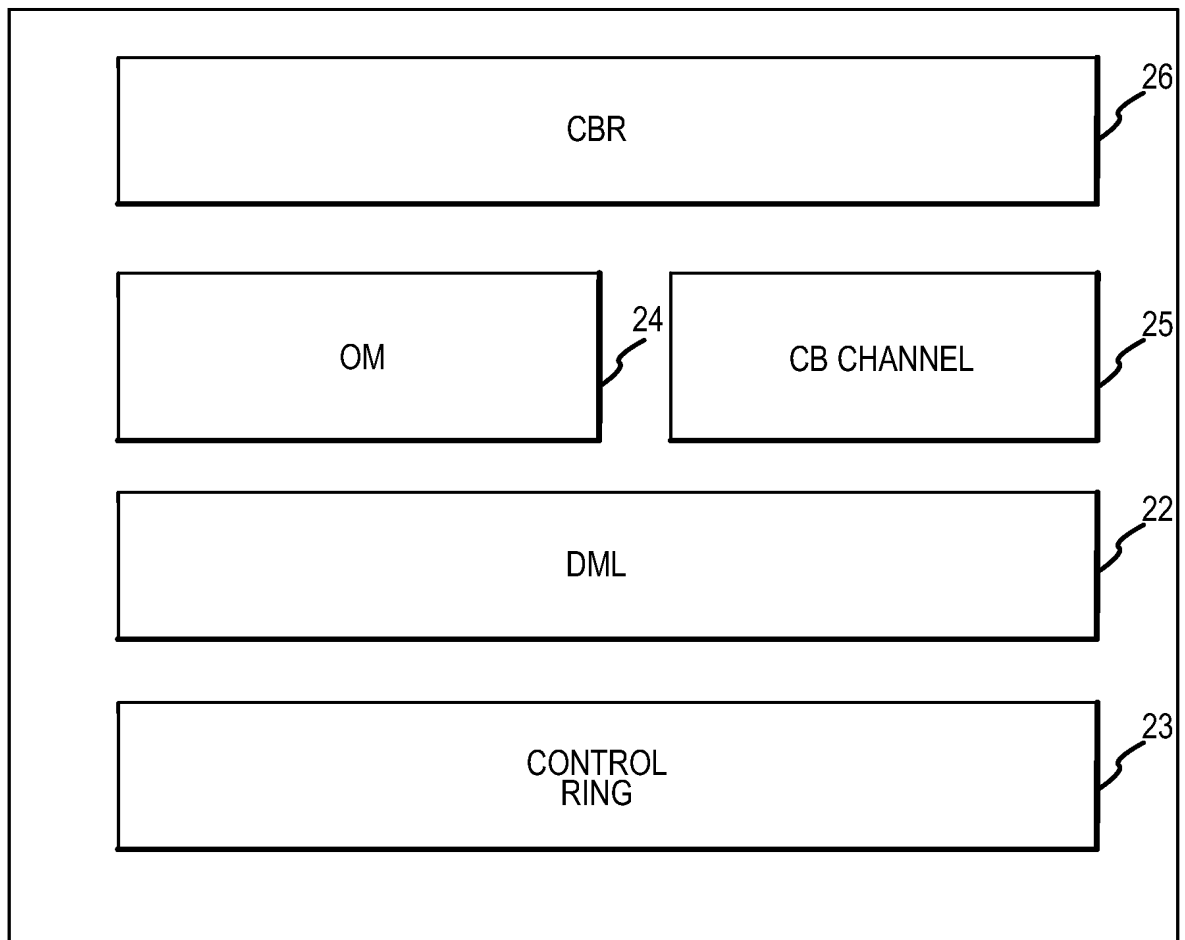


FIG.24

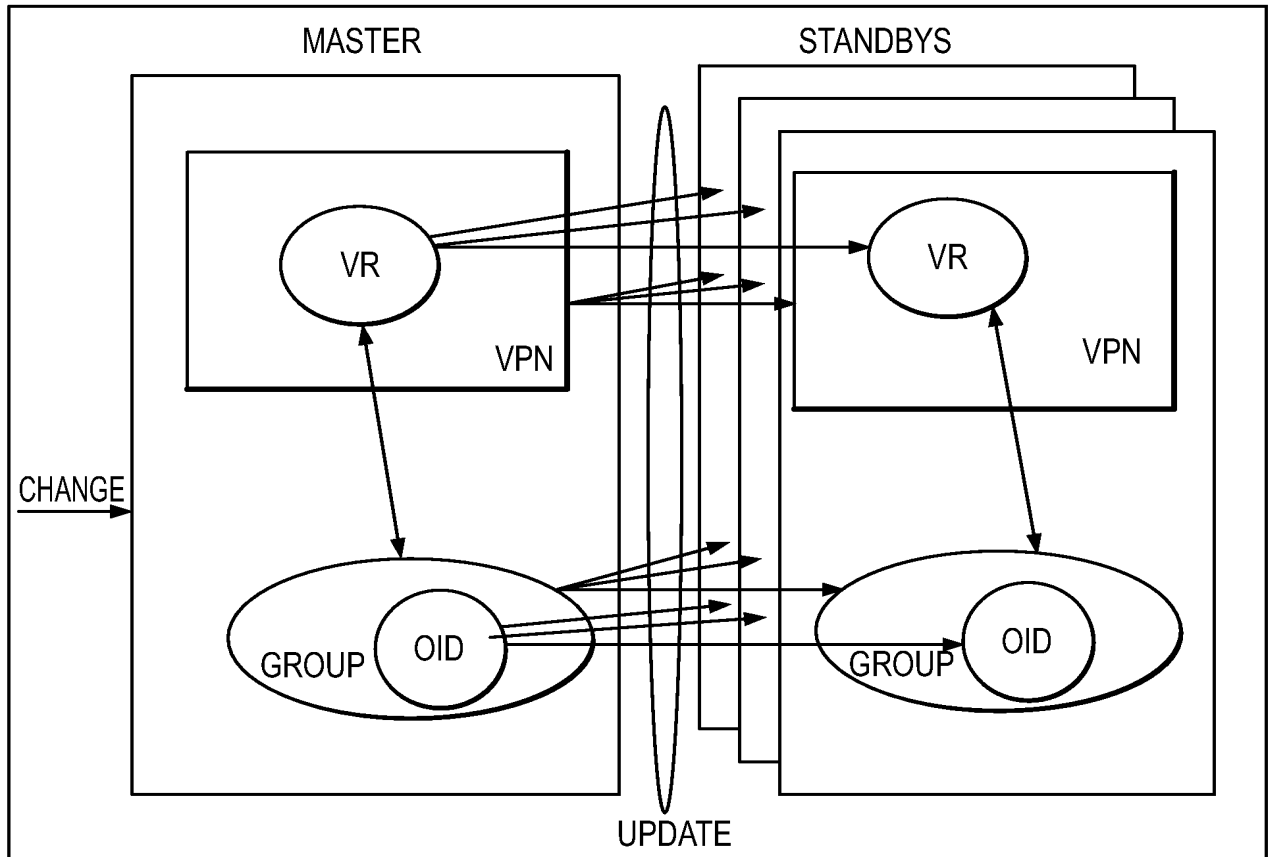


FIG.25

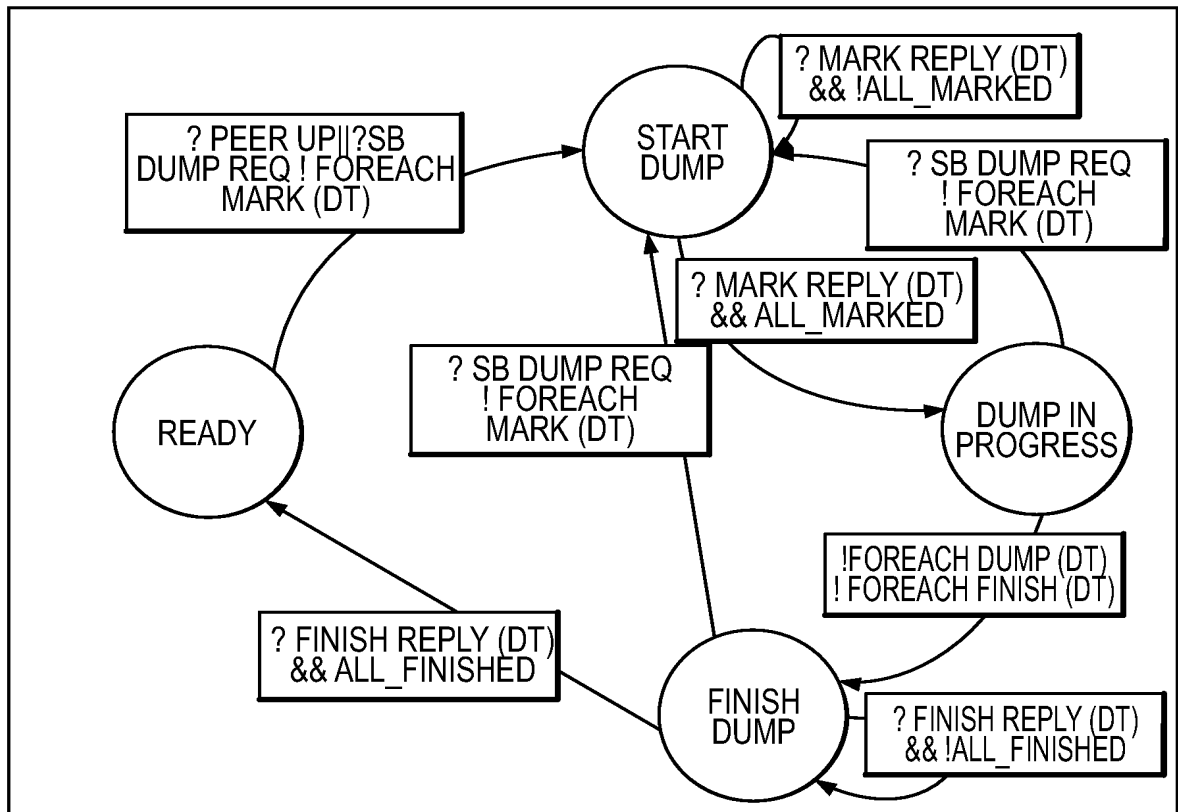


FIG.26

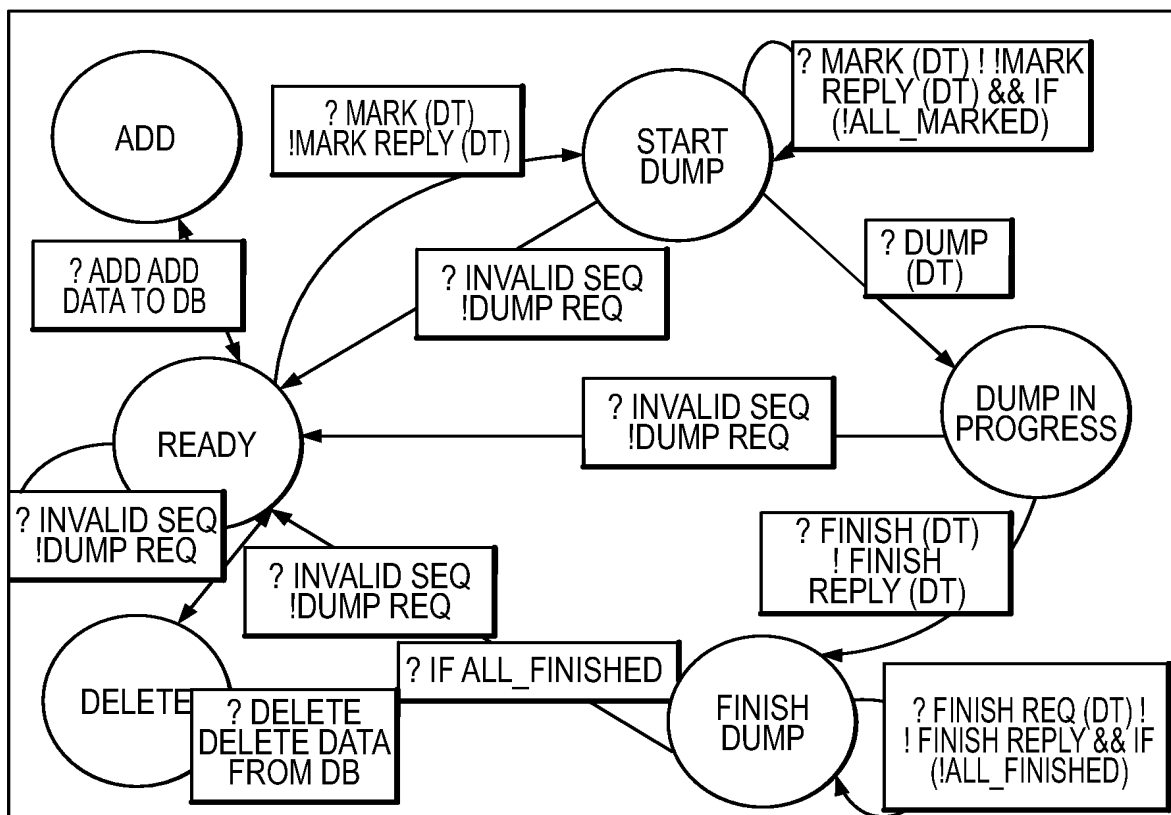


FIG.27